Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.4 No.1Ø July 1984

Shortly after we started using the Essendon primary school to hold our meetings, the principal, who used to come along because in those days pupils came in between 1Øam and noon to use the computers, saw how interested some of the children were in using the computers and decided it might be a good idea if the school got one. I lent the school a Series I (converted to 48 screen) Superboard, a 12" TV converted to direct video and an old cassette recorder.

Two teachers at the school started using the computer for their grades 5 and 6 maths classes, saw how it was helping the children and asked the school council for money for more computers. Over the next year we found four more 2nd hand Superboards for them to buy and they bought K Mart TVs and recorders to use with them.

A new principal came to the school about this time and after some initial doubt about the value of computers, sought donations of old typewriters so that the students could practice on the keyboards and so make better use of their time on the computers. It is not the aim of the school to produce computer experts but all students do some programming and most of the programs used, which are either maths drills or 'fill in the missing number or letter' type (no Space Invaders) are written by the more advanced students.

The teachers and the school council were so impressed by the benefit that the children are receiving from the computers that they decided to include grades 3 and 4 in the program. To do this the school has just purchased four Commodore 64s !!?? and a disk drive and is now the best computer equipped state primary school in the district.

The next meeting will be at 2pm on sunday 29th July 1984 at the Essendon Primary School on the corner of Raleigh and Nicholson Streets, Essendon.

The closing date for articles for the August newsletter is 1Øth August.

INDEX

APPLE SCREENS

by David Anear

The Apple video, in my opinion is the one feature which distinguished this computer from its competitors at the time of its release. Now outmoded in many respects by the latest computers to hit the market it has by virtue of the mountain of software written for it, become the industry yardstick (Yard = 914.4mm).

The Apple has three distinct video modes. A text mode that displays 24 lines of 40 characters, and two graphics modes. Lo-Res graphics, which occupies the same memory space as that reserved for the text page ($400-$800), has a resolution of 40 dots vertically. Each dot is large, being 7 by 8 pixals but can be displayed in any of 16 colours. Hi-Res graphics mode on the other hand has a resolution of 280 dots horizontally by 192 vertically, but can only display 6 different colours. (There are actually 8 colours including 2 whites and 2 blacks.)

Both graphics modes can be either full screen or a mixture of graphics and 4 lines of text at the bottom of the screen. This mode reduces the Lo-Res screen to 40 lines and the Hi-Res to 160 lines.

Each of the graphics modes has 2 distinct pages or screens, hardware set in memory. Each screen can be viewed separatly by setting a series of software switches residing in reserved memory. These are not real physical switches but switches that can be toggled by POKEing values to their reserved memory locations. These switches tell the video hardware to display either text or graphics, Lo-Res or Hi-Res, full screen or mixed text/graphics, and either page one or two.

All this sounds very nice, however, there are a couple of worms in the Apple! The two Hi-Res screens reside in memory locations $2000-$3FFF for page 1, and $4000-$5FFF for page 2; but Applesoft Basic starts storing its programs at $800 upwards. It does not take a genius to work out that after a little while your program is going to over-write the Hi-Res memory area. So you have to set the LOMEM pointer to store any Basic programs above the Hi-Res screen you wish to use.

If you are wondering why the Hi-Res screens are in such a CRAZY position, it's historical again as the first Apples had only 16k of Ram, and came with an integer Basic that stored its programs from the top of memory downwards. The Hi-Res screen was ideally placed for integer - but along came Applesoft.

For the sake of 2 chips (a lot of money in 1976), Apple's designers layed out their screen rather differently to what is considered normal today. There are $28 Hex bytes between the start of the first line of characters and the start of the next line. But the physical position of the second line is a third of the way down the screen instead of directly below the previous line as in any other computer. The upshot of this is you need a display map of the screen to help you if you wish to POKE characters to the screen directly (a real bind in Hi-Res). Fortunately Apple provide the capability of writing anywhere via cursor for Basic programmers and a small machine code routine for locating any start of line position for us machine code freaks.

Unlike most other machines which have Hi-Res screns, Apple only displays 7 bits of the 8 bits available. The missing MSB of each Hi-Res screen byte is used to determine the colours of the remaining bits on the screen. How do you get 8 different colours from ONE bit? (put your analyst on danger money!) I'll let you figure that one out for now, as it takes a bit of explaining.

Applesoft Basic has commands for plotting, line drawing etc built in, and most of the Hi-Res routines are callable from machine code. All in all the Apple is a truly remarkable machine for graphics considering its limited resolution, and with some of the software available on it, is the envy of most other manufacturers.

ANOTHER VERSION OF DIR

by King Corky

This is yet another mod to the very-much reworked Directory utility for OS65D. For those of you who still live in the dark ages, the few CompDos 1.2 commands can be removed.

The new enhancements give a display of slightly less than a screenful, with a prompt to get the rest of the directory. It then tells you the name and range of any spare tracks. The only stipulation placed on this last feature is that spare tracks should be created in the directory with the filename 'SPARExx'.

One of the many attractions of CompDos is its ability to create a directory entry at the same time as a program is saved to disk, ie. 'PUT Filename', without having to create the entry first, or conversly, to create a directory entry without having to specify a track range (just the required number of tracks). CompDos also allows us to RENAME any existing file without the need to run a special renaming program. This ability tends to make one a little slack regarding the amount of disk space left to hold any new programs that one may type in. The result of this slackness is that sometimes there is insufficient room on the disk to store the program we have just written.

My solution to this problem came about because I am a bit of a miser and tend to pack a lot onto a disk and have been caught enough times to want to do something about it, so now I make sure my work disks have full directories, whether the tracks are used or not. All unused tracks are entered in the directory with file names of SPARExx. I also vary the number of tracks allocated to each SPARE entry to give me scope for varying program sizes. Some of these SPARE tracks are regularly used for temporary storage of routines being developed, this means that I don't have to dream up and remember fancy file names for these routines.

To enable me to keep track of how many SPARE tracks have been allocated, I have included in the DIR, a subroutine that will pick up these SPAREs and list them at the bottom of the normal directory. This a lot quicker than staring googly-eyed at the directory on the screen, counting and writing down all those unused tracks that may be scattered throughout the disk. They are all listed at the end, with their track ranges.

What this means of course, is that if you want to make full use of this facility, you need to go back to pre CompDos days and make sure that directory entries are created before any work is done. With CompDos this is no real problem as when any SPARE track is used, you can just rename it with 'RN,SPARExx,newname'.

```
Compdos 1.2  NMHz    *   Volume - Work-1     *    Drive A
===========  ====        ===============          =======


File          Track Range                File          Track Range
-----------------------                  -----------------------
Work-1  ----   0  -  0                   OS65D3  ----   0  -  8
BEXEC*  ----   9  -  9                   NEWDOS  ----  10  - 10
DIR     ----  11  - 11                   DECHEX  ----  12  - 12
DIROLD  ----  13  - 13                   ASAM68  ----  14  - 17
SPARE1  ----  18  - 18                   PNTRAS  ----  19  - 19
SCDUMP  ----  20  - 20                   BUFSET  ----  21  - 21
PLOT*!  ----  22  - 22                   BACKUP  ----  23  - 26
SPARE2  ----  27  - 27                   RENUM   ----  28  - 30
GRAFIX  ----  31  - 31                   TESTJS  ----  32  - 32
SPARE3  ----  33  - 33                   CRAZY8  ----  34  - 37
SAUCER  ----  38  - 39                   FLASH   ----  40  - 40
CIRCKT  ----  41  - 42                   ZENER   ----  43  - 43
SEMF.S  ----  44  - 44                   SAUCR2  ----  45  - 46
DRSORT  ----  47  - 47                   SPARE4  ----  48  - 48
DISSAM  ----  49  - 50                   SPARE5  ----  51  - 51
WP2DIR  ----  52  - 53                   MAR2ST  ----  54  - 54
SPARE6  ----  55  - 58                   PARTY   ----  59  - 59
SOUNDS  ----  60  - 60                   SCHASE  ----  61  - 61
MATCH   ----  62  - 62                   NEWBEX  ----  63  - 63
DRSRT2  ----  64  - 64                   COL3.3  ----  65  - 65
SLOTS   ----  66  - 68                   COLRBA  ----  69  - 69
OLDBEX  ----  70  - 70                   SPARE7  ----  71  - 73
BADMP2  ----  74  - 75                   SCRCLR  ----  76  - 76

                There are 7 empty files.
                -------------------------

SPARE1  -   18 - 18              SPARE2  -   27 - 27
SPARE3  -   33 - 33              SPARE4  -   48 - 48
SPARE5  -   51 - 51              SPARE6  -   55 - 58
SPARE7  -   71 - 73
```

Program on page 12.

Superboard

July 1984.
NEWSLETTER OF THE OHIO SUPERBOARD USER GROUP, 146 YORK STREET, NUNDAH, 4012.

EXTENDED MONITOR ASCII DUMP by Derryl Cocks.

About a year ago, I attended an OSI User Group meeting in Wellington, New Zealand. There was an Eprom Programmer based on a 6800 available for use by members. The machine also would print out the Eprom contents in Hex and Ascii. If the Hex character was not a valid Ascii character, a full stop was printed in its place. Here is the routine for EXMON's Dump utility. The original EXMON at 0800 is assumed.

```
1000 CA       DEX            Come here from $0D0F
1001 F0 03    BEQ $1006      Dump ready?
1003 4C 04 0D JMP $0D04
1006 20 51 0C JSR $0C51      Yes, output two spaces first.
1009 20 51 0C JSR $0C51
100C A2 10    LDX #$10       Parse the current line again.
100E A5 DC    LDA $DC
1010 38       SEC
1011 E9 10    SBC #$10
1013 85 DC    STA $DC        Dump address Lo byte
1015 B0 02    BCS $1019
1017 C6 DD    DEC $DD        Dump address Hi byte
1019 B1 DC    LDA ($DC),Y
101B 20 27 10 JSR $1027
101E 20 36 0B JSR $0B36      Dump ready?
1021 CA       DEX
1022 D0 F5    BNE $1019      Line ready?
1024 4C F2 0C JMP $0CF2
1027 29 7F    AND #$7F       Mask off high bit for Ascii
1029 C9 7F    CMP #$7F       Delete character
102B F0 04    BEQ $1031
102D C9 20    CMP #$20       Space character
102F B0 02    BCS $1033
1031 A9 2E    LDA #$2E       Full stop
1033 20 EE FF JSR $FFEE      Print it
1036 60       RTS
```

To splice this into the EXMON, you also need to change three bytes from $0D0F 20 00 10. You will now need to save the EXMON back to tape. EXMON now occupies 0800 - 1036 Hex.

- -

PLAY - BY - MAIL OFFERS BIG ADVENTURES ON A COMPUTER

I suppose all of us have played Hamurabi, or it's big brother. King, on a computer. These were very early games of adventure simulation, written long before the days of even the most simple video graphics. Usually, they ran on teletypes. The idea was to survive a ten year or longer term of office while running a country. As ruler, you had to feed the population, buy and sell land, keep pests under control and etc. Your computer would issue a yearly report. Hazards included uprisings if too many of the people starved, being deposed if foreigners exceeded locals, pollution keeping the tourists away, and bankruptcy. Play-By-Mail games take the complexity of these simulations to an extreme. Let's look at one which arrived in my mailbox recently. The game is called "Return from Sirius"

— SUPERBOARD —

Scenario: You are the ruler of an industrialised planet far from Earth. Earth has controlled a galactic empire of hundreds of stars for thousands of years. As the empire grew, it became increasingly difficult to rule, and finally, the government decreed that any planet found with a space vehicle capable of using hyperspace would be destroyed. The great earth trading ships plied the stars for millenia, accompanied by the space patrol, which enforced the hyperspace law. As time went on, the visits became less and less frequent and the last was some 800 years ago. The Sirians offer you a gift of four starship hulls and motors. The offer is also made to other planets. The risk is annihilation, the reward, possible control of the galaxy. You, and the rulers of eight other planets accept the offer.

Game: To win the game, you must reach a wealth of 20,000 Credits, own 1/3 of the galaxy, or control Earth. A typical game takes 25 turns, and as games are on a two week turnaround basis, that means a total of a year. Each turn represents a year of gametime. A turn consists of a list of orders to your starships and planets. You can also send diplomatic messages to other players you encounter. Basically, you go exploring, build up your ships, mines, industry, and planetary defence. Your ships will encounter many planets, which lacking opposition, you claim and name for your empire. You load your ships with a selection of ore types which can be used by your industry to increase production. Soon, your expanding empire will overlap those of other players, and you can trade or do battle.

The Return from Sirius document goes on to give details of how you earn Credits, and the way you issue orders. All this is fed into a computer which then "moderates" the game, printing out your status after each turn. Along with the status report, you receive a general newsletter plus any diplomatic letters. Whether a player with a computer would be at an advantage is hard to estimate. I am sure you will agree that the game would be quite difficult to code, particularly for the moderating task.

If you are interested in playing in the game, or getting more information about it, write to The Missing Tiger, GPO Box 286C, Hobart, TAS 7001.

- -

BASICODE PROTOCOL continued from last month.

Line Numbers:	1 - 10	Free to designate
	10 - 100	Program Identification
	100-500	Array function definition, and initiation.
	500-1000	Machine independent subroutines
	1000-10000	Main program
	-20000	Program dependent subroutines
	-25000	Machine dependent sub-programs,reading files.
	-30000	Data
	30000-	Rules, purpose of program, operation REMs

Special attention should be given to lines 10-100.

```
10 REM title of program
20 REM name, address of author
30 REM machine type, language version
40 REM data and edition number
50 REM any extra essential hardware needed
60 REM references
70-90 REM variable names, dimensions
-100 REM dialect dependable
```

All names of variables may consist of up to a maximum of 2 significant characters, of which the first must always be a letter. Combinations with X, Y, and Z should not be used in the main program. X and Y are reserved for sub

programs, while combinations with Z are reserved for graphic routines.
$ indicates a string variable
% indicates an integer variable (not used in OSI)
It is recommended that no more than six decimal places are used.

The first two letters of a variable should not be the same as the first two letters of any of BASIC's reserved words. This is a tall order as new words are being used all the time. The complete list I have is too large to print here, and there are even more BASIC words used in the latest colour computers.

Finally, the cassette label should contain the following information:-
Program Name, Author's name and address, Computer type, program size, and the word BASICODE.

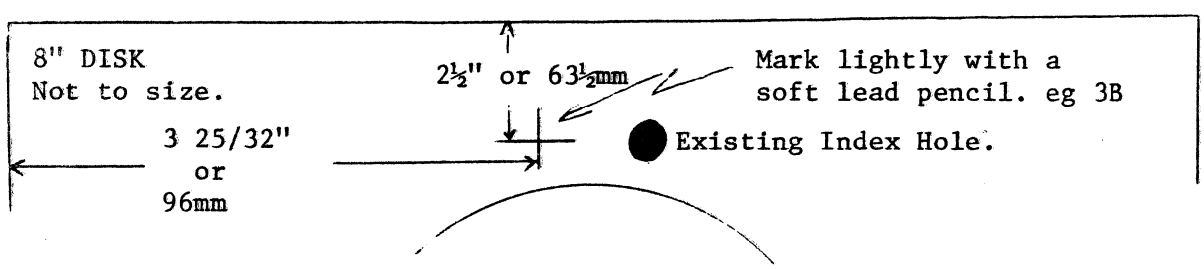
TURN FLOPPIES INTO FLIPPIES

The Apple disk article last month suggested this item on how to doubleside 8" and 5¼" singlesided disks. To do this neatly, you need an Alco Hand Hole Punch (single hole and a low profile), plus a good ruler. The punch is available from Woolworths and costs a little over a dollar. Disk makers warn that disks should not be turned over. They argue that dust and loose oxide that is trapped by the tissue material in the case is dislodged if the disk is made to rotate in the opposite direction, and damages the magnetic surface. I'd be more likely to believe this if someone other than the disk manufacturers confirmed it. If untrue, it would effectively halve sales.

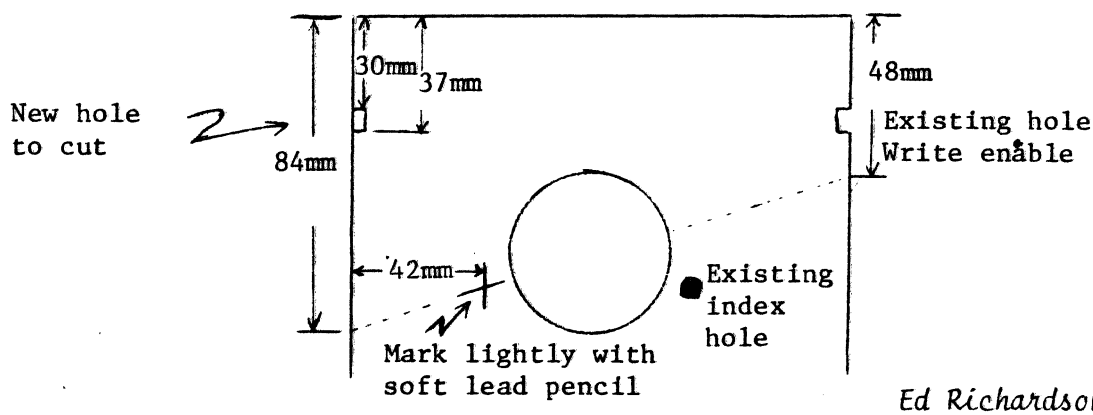
In any case, if you have a disk with one or more bad tracks, you have little to lose. All disks are manufactured doublesided, the second side being sometimes untested.

8" DISKS: Mark both sides as in diagram below. With clean,dry fingers, gently insert punch between disk and case, and centre the + in the hole of the punch. Punch the hole, ejecting the dump outwards. Turn the disk over and do the other side. The job is all done!

If you want a write protect notch, mark position using another 8" disk.



5¼" DISKS: Mark the position of the write enable notch as shown below. Draw the line to find the position of the index hole. Punch as described for 8" disks. The write enable notch MUST be cut on a 5¼" disk.



Ed Richardson.

UNDERSTANDING FORTH Part 1

by Ray Gardiner

All the examples used in this series of articles are written in fig Forth and some of the examples assume you are using disk, Forth Tools or Technical Products fig Forth for the OSI and Rabble.

1. INTRODUCTION

: purpose

This series of articles is intended to open a window into the world of Forth. This will be accomplished largely by presenting examples of Forth code and dissecting same to gain an insight to the inner workings of the language.

: disclaimer

You will find the best way of learning Forth is to do your learning at the keyboard, the highly interactive nature of the language makes this process somewhat faster than other languages. (Be adventurous and experiment a lot, what could you lose, (maybe a disk or two.))

:references

Starting Forth by Leo Brodie (Polyforth)
Forth Tools by Martin Anderson (Forth 83)
Forth Dimensions (bi-monthly newsletter)
Forml Proceedings (available from MV press **)
Rochester Conference Proceedings (available from MV press **)

** Mountain View Press, inc.
PO Box 4656
Mountain View
CA 94Ø4Ø U.S.A. Ph. 415 961 41Ø3

Some reference literature is desirable as you will learn quickly from examining other programmers code and adapting various words from other applications.

: glossary

Forth has its own jargon like everything else, so here is a list of the words that we will use in the introduction. Read carefully and digest, we will ask questions later!

DICTIONARY	The linked list comprising the Forth system.
EXECUTE	Execute a WORD, (run), carry out the process.
WORD	Used to describe one dictionary entry, everything in Forth is a WORD.
VOCABULARY	A subsection of the dictionary separately linked.
DEFINITION	The sequence of WORDS used to define a new WORD.
STANDARD	The EXECUTION and DEFINITION of a complete Forth system set of WORDS is controlled by various international standards. There have been several, fig Forth, Forth 79, Forth 83, none are obsolete but simply assist portability.

: example one

The first program in all languages is a program to put some text on the screen, in Forth it looks like this.

```
: GREET ." Hello World " ;     <press cr>
OK                        <Forth responds with OK>
```

We have now added a new word to the dictionary and may execute it by simply typing the name of the word.

```
GREET                  <press cr>
Hello World OK
```

Note that there is no cr/lf appended to the output, if a cr/lf is required it must be invoked with the word CR which will alter our definition slightly.

: example two

```
: GREET CR ." Hello World " CR ;
```

Typical execution would be.

```
OK
GREET
Hello World
OK
```

Note All words in Forth need to be delimited (separated) by spaces with no exceptions, this applies to " since " is not a Forth word, merely a delimiter.

Several new WORDS have been introduced in the examples presented, these are.

- : Pronounced "colon". Tells Forth that a new definition is coming, start compiling a new dictionary entry.
- ." Pronounced "dot quote". This word scans ahead in the input stream looking for the closing delimiter " . Every ." must have a closing " . The string found is moved into the dictionary as part of the WORD being defined. At execution time the ." will output the string to the terminal.
- ; Pronounced "semi colon". Switches Forth back into the interpret state, that is, finish the definition and make it available.

Almost all of Forth is written in Forth, this gives us a perfect means of describing words and their behaviour in detail. Beginners may skip this section if they desire.
The definition of a colon is as follows.

```
: :  ?EXEC  !CSP  CURRENT @ CONTEXT ! CREATE ] ;CODE IMMEDIATE
      IP->RS   W+2->IP  JMP-NEXT
```

?EXEC	Error if found in compile state
!CSP	Save stack position.
CURRENT	Vocabulary first searched when interpreting.
CREATE	Build name into dictionary and search.
]	Enter compilation state.
;CODE	Rewrites code field pointing to machine code.

The code immediately following the ;CODE is written in machine code for the processor concerned.

IP->RS	Push the Interpretive Pointer on return stack. The code for the 65Ø2 to do this is LDA IP+1 PHA LDA IP PHA
W+2->IP	Step the code pointer to the next word and nest down one level by moving it to become the new IP CLC LDA W ADC #2 STA IP TYA ADC W+1 STA IP+1
JMP-NEXT	Re-enter the virtual machine instruction fetch.
NEXT	All Forth words terminate by looping back to NEXT, more detail on NEXT when we look at the nucleus.

The definition of semi colon follows

```
: ;     ?CSP COMPILE ;S  SMUDGE [COMPILE] [  ; IMMEDIATE
```

?CSP	Verify stack position.
COMPILE	Compile the execution address of following.
;S	Unnest system back up a level.
SMUDGE	Toggle a bit in the name field so that word just defined can be found.
[COMPILE]	Force immediate word following to compile.
[	Stop compilation enter interpret state.
IMMEDIATE	Toggle the precedence bit in the name field so that the word just defined will execute even while machine is in compile state.

The definition of ." involves two definitions, one for the compile version and one for interpret. As such ." is referred to as "state smart". Forth-83 has been rigorously purged of state smart words.

```
: ."   ASCII "   STATE @  IF  COMPILE (.")
                                WORD HERE C@ 1+ ALLOT
                           ELSE  WORD HERE COUNT TYPE
                           THEN ;  IMMEDIATE

: (.")  R COUNT DUP 1+ R> +>R TYPE ;
```

ASCII converts the following character to hex, this is then used by WORD to ENCLOSE the input stream and move the string to HERE which is then TYPEd if interpreting or simply stepped over by ALLOTing dictionary space if compiling.

The definition of of the run-time (.") involves some fancy footwork with the return stack in order to step over the string.

R	Copies the top of return stack.
COUNT	Leave the address and character count.
DUP	Copy the character count.
1+	We want the address just following string.
R>	Current address.
+	Add character count +1.
>R	Back to return stack.
TYPE	Output string.

: conclusion

We have introduced the : and ; and shown some examples using ." and " , now we will find out who was paying attention.

EXERCISES

1. Write a Forth word which will return the surname of a friend when you enter the first name.

2. Write a Forth word to type a Telephone number in response to a name. eg COMPSOFT 428-5269

3. Write a language translater to generate literal translation of some French or German text into English. HINT... : MERCI ." THANK YOU " ;

If you wish to check your answers then read next month's article or come along to a KAOS meeting and corner a Forth type.

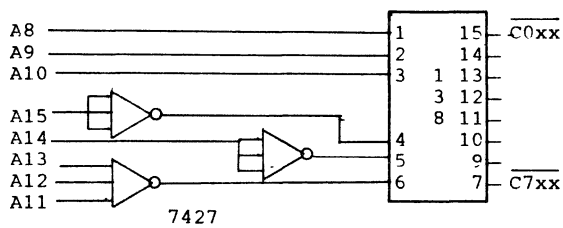
..... till next month, Ray Gardiner.

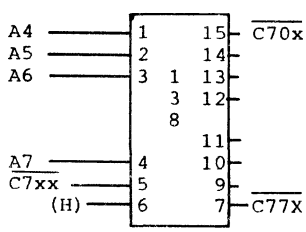
SIMPLE I/O DECODER

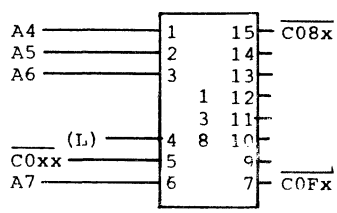
by Mark Howell

Some time ago an attempt at I/O memory map standardization for Ohio computers was published (see KAOS V1 No6 page 2). Unfortunatly only the $CØxx memory area was defined and anyone with a 505 (C2) or 610 (C1) floppy disk board will know that these boards are only partially decoded and write all over $CØxx.

Included here are three simple decoder circuits that may be of use to hardware hackers for decoding I/O locations in the $CØØØ-$C7FF memory area. The first circuit decodes to a single 'page' in this memory block while the second and third circuits decode into groups of sixteen locations suitable for VIA's etc. As an aid to I/O standardization maybe each 'page' in the $CØØØ-$C7FF memory area could be defined as a USER PORT numbered from Ø to 7. (See hardware memory map.)








```
.............................
.                           .
. OSI HARDWARE MEMORY MAP   .
.                           .
.    FOR THE C1 AND C2      .
.                           .
.............................
$0000-$1FFF END 8K RAM
$0000-$3FFF END 16K RAM      Compiled by
$0000-$5FFF END 24K RAM      Mark Howell
$0000-$7FFF END 32K RAM
$0000-$9FFF END 40K RAM
$0000-$BFFF END 48K RAM
$8000-$9FFF ROM ASM 65 ASSEMBLER (K.A.O.S. V4/N2 P10)
$8000-$8FFF ROM W/P 6502 (K.A.O.S. V4/N7 P2)
$8000-$87FF ROM TOOLKIT
$9000-$97FF ROM BASIC 5 (K.A.O.S. V3/N5 P4)
$A000-$A7FF ROM MICROSOFT BASIC 1 (K.A.O.S. V3/N9 P5)
$A800-$AFFF ROM MICROSOFT BASIC 2 (K.A.O.S. V4/N3 P10)
$B000-$B7FF ROM MICROSOFT BASIC 3 (K.A.O.S. V3/N3 P3)
$B800-$BFFF ROM MICROSOFT BASIC 4 (K.A.O.S. V2/N10 P4)
$C000-$C0FF FLOPPY DISC PORT - 505 & 610 BOARD
$C000-$C003 PIA FLOPPY DISC CONTROLLER
$C004-$C007 PIA 1 USER I/O (K.A.O.S. V3/N9 P7)
$C008-$C00B PIA 2 USER I/O - TASKER BUSS
```

$C00C-$C00F PIA 3 USER I/O - TASKER BUSS
$C010-$C013 ACIA FLOPPY DISC CONTROLLER
$C014-$C017 SOUND PORT - RABBLE BOARD (K.A.O.S. V3/N10 P9)
$C018-$C01B PORT USER - RABBLE BOARD
$C01C-$C01F PORT USER - RABBLE BOARD
$C020-$C02F CLOCK GENERATOR - RABBLE BOARD
$C030-$C03F VIA 1 USER I/O
$C040-$C05F VIA 2 USER I/O - TASKER BUSS
$C060-$C07F VIA 3 USER I/O - TASKER BUSS
$C090-$C0FF NOT ALLOCATED (APPLE I/O SLOTS 0-7)
* * * * * USER PORT I/O HARDWARE * * * * *
RS-232 SERIAL PORT
CENTRONICS PARALLEL PORT
CALCULATOR (K.A.O.S. V1/N11 P6)
SOUND GENERATOR (K.A.O.S. V1/N12 P8)
SPEECH SYNTHESISER (K.A.O.S. V2/N12 P5)
ANALOGUE/DIGITAL CONVERTER (K.A.O.S. V3/N6 P11)
EPROM PROGRAMMER (K.A.O.S. V2/N8 P2 & V3/N9 P4)
EPROM PROGRAMMER (K.A.O.S. V3/N3 P7 & V4/N3 P12)
EPROM PROGRAMMER (K.A.O.S. V4/N6 P11)
TIME CLOCK (K.A.O.S. V4/N1 P4 & V4/N5 P8)
CP/M BOARD (K.A.O.S. V4/N7 P3)
$C100-$C1FF PORT 1 USER (SERIAL PORT???)
$C200-$C3FF LATCH - B.W. VIDEO BOARD (K.A.O.S. V3/N12 P7)
$C200-$C2FF PORT 2 USER
$C300-$C3FF PORT 3 USER
$C400-$C5FF PORT - B.W. VIDEO BOARD (K.A.O.S. V3/N12 P7)
$C400-$C4FF PORT 4 USER (6532???)
$C400-$C47F RAM 6532 - RABBLE 65 (K.A.O.S. V3/N3 P6)
$C500-$C5FF PORT 5 USER (6845???)
$C600-$C6FF PORT 6 USER (PARALLEL PORT???)
$C700-$C7FF 16 PIN I/O BUSS (K.A.O.S. V3/N2 P8 & V4/N4 P8)
$C800-$CFFF RAM USER - RABBLE BOARD (K.A.O.S. V4/N4 P5)
$D000-$D7FF RAM VIDEO - C2
$D000-$D3FF RAM VIDEO - C1
$D400-$D7FF RAM COLOUR - C1
$D800-$DEFF LATCH - TASAN VIDEO BOARD (K.A.O.S. V3/N11 P9)
$D800-$DBFF LATCH SCREEN COLOUR SOUND - C1
$DC00-$DFFF PORT KEYBOARD - C1
$DE00-$DEFF LATCH SCREEN COLOUR SOUND - C2
$DE80 LATCH - S.E.K. (K.A.O.S. V3/N7 P5)
$DF00-$DFFF PORT KEYBOARD - C2
$E000-$EFFF ROM USER - RABBLE 65
$E000-$E7FF RAM COLOUR - C2
$E000-$E7FF ROM EX. MONITOR PLUS - C1 (K.A.O.S. V4/N2 P2)
$E800-$EFFF ROM EX. MONITOR (K.A.O.S. V3/N12 P2)
$E800-$EFFF ROM COMPDOS 1.2 - C2 (K.A.O.S. V3/N6 P10)
$F000-$F0FF ACIA CASSETTE TAPE - C1
$F700-$F7FF PIA - 505 BOARD
$F800-$FFFF ROM CEGMON
$F800-$FBFF ROM DABUG - C1 (K.A.O.S. V3/N8 P2)
$F800-$FBFF ROM DABUG 3J - C1 (K.A.O.S. V4/N1 P11)
$F800-$FBFF ROM RESMON - C1 (K.A.O.S. V4/N6 P6)
$FB00-$FBFF UART - 430 BOARD
$FC00-$FCFF ACIA CASSETTE TAPE - C2
$FC00-$FCFF ROM FLOPPY BOOTSTRAP - C1
$FD00-$FDFF ROM KEYBOARD (K.A.O.S. V3/N8 P8 & V3/N9 P8)
$FE00-$FEFF ROM 65V MONITOR
$FF00-$FFFF ROM RESET BASIC SUPPORT
$FF00-$FFFF ROM RESET - 505 BOARD (FLOPPY BOOTSTRAP???)

```
10 REM  Directory Utility For COMPDOS 1.2 (OS-65D 3.2)  NMHZ
15 :
20 REM  From PEEK(65) vol 4,number 4,to print volume ID in
30 REM  directory heading.Vol ID must be 1st entry with track
40 REM  range of '00-00'
50 :
60 REM  Modifications by R.Cork as at 3/5/84
70 :
80 NF=0:B=0:C=0:Z=0:SC=0
90 PN=11897
100 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
110 DV=2:DISK!"CL"
120 PRINT"Directory for which drive (A,B,C,D) ";:GOSUB140
130 GOTO155
140 DISK!"GO 252B":A=PEEK(9815):RETURN
155 IFA=13ORA=65THENA$="A":GOTO170
156 IFA=66THENA$="B":GOTO170
157 IFA=67THENA$="C":GOTO170
158 IFA=68THENA$="D":GOTO170
160 IFA$("A"ORA$)"D"ORA$=""THENA$="A"
170 DISK!"SE "+A$
180 DISK!"CL":PRINT:PRINT
185 PRINT"Send to the Printer instead of the Screen ?"
190 GOSUB140:IFA=13THENDV=2:GOTO210
200 IFA()78THENDV=4
210 DISK!"CL"
220 PRINT#DV
260 REM
270 REM PRINT A DIRECTORY OUT
280 REM
290 PRINT #DV,"Compdos 1.2  NMHz     *   Volume - ";:GOSUB660
300 PRINT #DV,"    *     Drive ";A$
305 PRINT #DV,"===========  ====        ===============         ======
320 PRINT #DV:PRINT #DV:PRINT #DV
330 PRINT #DV,"File         Track Range";
340 PRINT #DV,TAB(35)"File         Track Range"
350 PRINT #DV,"-----------------------";
360 PRINT #DV,TAB(35)"-----------------------"
370 DISK ! "CALL 2E79=08,1"
380 GOSUB440
390 DISK ! "CALL 2E79=08,2"
400 GOSUB440
410 PRINT#DV:GOSUB 1000
420 POKE741,76:POKE750,78:POKE2073,173
425 PRINT:PRINT:PRINT TAB(12);"( Press RETURN for BEXEC* )"
427 PRINT:PRINT TAB(17);"( On Drive - A )"
430 GOSUB140:IFA=13THENDISK!"SE A":RUN"BEXEC*
435 END
440 REM READ DIRECTORY OUT OF BUFFER INTO ARRAYS
450 REM
460 FOR I=PN TO PN+248 STEP 8
470 IF PEEK(I)=35 THEN NF=NF+1 : GOTO 590
480 N$=""
490 C=C+1:AV=0
500 FOR J=I TO I+5
510 N$=N$+CHR$(PEEK(J))
520 NEXT J
525 S$(AV)=N$:GOSUB 900
530 IF C/2=INT(C/2) THEN570
535 AV=AV+1:SC=SC+1
536 IF DV=4 THEN 540
538 IF SC=12 THEN GOSUB 2000
540 PRINT #DV,N$;TAB(8);"-----";TAB(13);FNA(PEEK(I+6));
550 PRINT #DV,TAB(17);"-";TAB(18);FNA(PEEK(I+7));
560 GOTO 590
570 PRINT #DV,TAB(35);N$;TAB(43);"----";TAB(48);FNA(PEEK(I+6));
580 PRINT #DV,TAB(52);"-";TAB(53);FNA(PEEK(I+7))
590 POKE 55068,139:NEXT I
600 RETURN
610 :
620 REM   Reads 1st entry in directory.If track range is 00-00
630 REM   then entry is a volume ID.
640 :
650 :
660 DISK!"CA 2E79=08,1"
670 FOR I=PN+6 TO PN+7
680 IF PEEK(I)()0THENPRINT#DV,"(No ID)";:RETURN
690 NEXT
700 N$=""
710 FOR I=PN TO PN+5
720 N$=N$+CHR$(PEEK(I))
730 NEXT
740 PRINT#DV,N$+" ";
750 RETURN
900 :
920 IF LEFT$(S$(AV),5)()"SPARE" THEN RETURN
930 Z$(Z)=S$(AV)
940 X1(Z)=FNA(PEEK(I+6)):X2(Z)=FNA(PEEK(I+7))
950 Z=Z+1
990 RETURN
1000 PRINT#DV:PRINT#DV,TAB(14);"There are";Z;"empty files."
1010 PRINT#DV,TAB(14);"-----------------------":PRINT #DV
1020 FOR Q=0 TO Z
1025 B=B+1
1026 IF X1(Q)=0 OR X2(Q)=0 THEN 1050
1030 IF B/2=INT(B/2) THEN 1048
1040 PRINT#DV,Z$(Q);"  -  ";X1(Q);"-";X2(Q);
1045 GOTO 1050
1048 PRINT #DV,TAB(32);Z$(Q);"  -  ";X1(Q);"-";X2(Q)
1050 NEXT Q
1060 RETURN
1999 :
2000 PRINT:PRINTTAB(14);"Press RETURN to continue."
2010 GOSUB 140
2020 IF A=13 THEN PRINT::PRINT:RETURN
2030 GOTO 2010
```

KAOS QUEENSLAND MEETING 17/6/84

Attendance: 8 Computers: 5 (all working)
This was the first time I had advertised a meeting three weeks ahead in the Newsletter and not phoned anyone. Ringing 40 people was always a lot of work for me, taking up about 3 nights usually.
Anyway, only 8 turned up, and 3 of them had BBC's. I'm thinking of changing the name to BBC Users Meetings. The weather outside was wet, windy and cold.

Robin Wells, John Froggatt, and Paul Brodie had the Beebs. In the case of Paul, it was a short tenure as he was fitting drives to a school's machine. Paul reported that the Rom based DOS was a bit like CP/M to use. As John had fitted an 80 track drive, he had built up a RS423 lead for the purpose of transferring programs. It all worked very well. John also demonstrated his latest OSI effort, FROGMON. More on this CEGMON adaption in future issues of SUPERBOARD. Robin showed off a neat light pen, drawing lines, rectangles, and other figures on the screen. John demonstrated his latest graphics Rom.

Doug Robinson had purchased a Chinon (made in Japan) 40 track slimline drive from Energy Control for $212 incl. tax. This featured a neat printed circuit motor - a thing of wonder - and was beautifully built and silent in operation.

Ian Mackenzie was still exploring Hexdos in one of the school's stock standard OSI C1P-DF. Being a DOS which works with the Basic-in-Rom, it still left 29k of free Ram in a 32k system. Paul was able to demonstrate some of the finer points, including the line editor which works with Synmon.
Ian had been experiencing problems with some of the school's computers, OSI and Microbee. Apparently some bright little basket had discovered that computers don't work so well after you insert the video and cassette leads into the power lead for the cassette recorder. Apart from the expense of repair, the dangers of electrocution were very real.
Bob Best suggested that as the cassette recorders were never used as portables the power lead could be glued into place using silicon rubber or other suitable adhesive.

Nev Villiers (C8-DF) has an interest in sharemarket analysis, and would be pleased to correspond with any other user with similar interests. His address is

My thanks to Clive Harman and others who rang to offer a cure for the WP6502 problems last mentioned in this column.

Next Meeting: ********** A U G U S T 5 t h ********** SEX SEX SEX ****
(Now maybe, members will notice)

Ed Richardson.

Some months ago there was a request for help from a member who was using a Tandy printer and having problems with double line feed. Two members sent fixes so we are printing both of them in the order we received them. The second one will be in next month's newsletter. These routines should work with any printer which provides an automatic line feed.

DOUBLE LINE FEED PROBLEM AND TANDY DMP1ØØ (LINE PRINTER VII)

by Wolf Horn

With the Tandy computers the LPRINT command removes the line feed from the computer, the printer then generates its own line feed and everything is normal. When you connect the printer to an Ohio computer they both generate a line feed, hence the double spacing.

I have developed a routine (software patch) for use with BASIC and another for use with the Extended Monitor, they can easily be included with any program and they take up only a few bytes.

Part 1: Under BASIC - This problem can easily be cured by a simple and short software patch. It can be loaded into the computer after it is first switched on or it can become part of the program itself. It works during program listings as well. During a program run, each time you require a printout all you do is POKE 570,0 (ASCII NULL) and after a printout (or each printout within a program) you POKE 570,10 (ASCII LF restored) to allow normal screen operation again.

This program is stored in an unused (by BASIC) part of page 2. I started the routine at location $0235 since this is near the beginning of the free space and more importantly it places the byte we need to change at $023A or decimal 570 (an easy number to remember). If the Break key is pressed then the vector at $021A and $021B will have to be changed to point to our routine again. In BASIC you POKE 538,53 and 539,2.

A sample program below shows how to use the patch within a BASIC program. Between lines 1 and 10000 you insert POKE 570,0 before each printout and POKE 570,10 after each printout to restore the screen to normal, along with the usual POKE517,xx (SAVE on/off).

```
1 GOSUB 10000
2 POKE 517,255:REM SAVE ON
10 REM TEST PROGRAM
20 FOR L=1TO3:PRINT"DOUBLE SPACING WITH POKE 570,10":NEXTL
30 POKE570,0:REM REMOVE LF NOW
40 FOR L=1TO3:PRINT"SINGLE SPACING NOW WITH POKE570,0":NEXTL
50 POKE570,10:REM RESTORE LF IN PATCH
60 POKE517,0:REM SAVE OFF
70 END
10000 FOR L=1TO11:READ A:READ B
10010 POKE A,B:NEXT
10015 RETURN
10020 DATA 538,53,539,2:REM CHANGE O/P VECTOR ON PAGE 2
10030 DATA 565,201,566,10,567,208,568,2,569,169:REM = PATCH
10040 DATA 570,10,571,76,572,105,573,255
```

PART 2: Machine Language Version -

Under Extended Monitor the routine needs to be loaded in a different way, one data item needs to be changed as well. The Ext Mon uses the LF command and we therefore have to remove the carriage return instead. As well as this you can not POKE to the locations $021A and $021B - so it is better to load the routine shown below to say $1000 and then type G1000. The Ext Mon prompt will reappear and now if you want anything printed out just type S for save, the result is a printout of say a disassembled section of memory without the nasty double line feeds.

```
1000 A935   LDA #$35
1002 8D1A02 STA $021A
1005 A902   LDA #$02
1007 8D1B02 STA $021B
100A A209   LDX #$09
100C BD1610 LDA $1016,X
100F 9D3502 STA $0235,X
1012 CA     DEX
1013 D0F7   BNE $100C
1015 00     BRK
1016 C90D   CMP #$0D
1018 D002   BNE $101C
101A A900   LDA #$00
101C 4C69FF JMP $FF69
```

To get C/R back to normal (on screen)
Under Ext Mon type Shift P
This gives you the @ symbol
Now enter the address 0236
The Ext Mon prints 0236/0D
Now type 00 for normal screen.

THE MEETING WAS KAOS

by King Corky

Today, being kiddies day, saw the return of the Turtle and the Speaking Computer, both owned and operated by the same dedicated hacker, viz Jeff Kerry.

Compsoft now have the I/O card ready for their CP/M system and this will enable/allow any of the OSI/Rabble or look-alikes to run the CP/M cards, and have slots left for printers, modems etc. Paul Dodd also showed us a micro-floppy drive and disk, that's the 3 1/2" floppy system, in 80 track single sided format, (80 track double sided 1Meg is also available). These drives are compatable with current 5 1/4" drives, draw only 200-300 mA, and the best part, cost is under $250 inc. tax.

David Anear and Ray Gardiner again had the 68000 DTACH board on display, but this time the Rabble/Apple interface card had been attached. The card address required it to be plugged into the number 3 slot in the Apple, but on David's machine this slot was taken up by some other necessary hardware and therefore we could not see it running in full colour. A shame, because it would have been marvellous to watch this hard-to-come-by board in action.

Ray Gardiner demo'd a Forth based modem routine that has online directories, STD charge rates, commands, mainframe time sharing info and numerous other goodies that would make using a modem a breeze. David said a 65U version with similar facilities is available. He also mentioned that Frank Nicolls has a 65U Browse facility with a nibble editor. Because of copyrights etc, royalties of $35 are payable on this and from what I have been told its worth every penny. Ray Gardiner has upgraded his Forth editor, which now gives insert/delete keys and full cursor editing of a Forth screen.

Well that's all for now. Bye.

FOR SALE

ICL Termiprinter. Prints 10 characters per inch on standard 11" x 12 7/8" tractor feed paper. Up tp 118 characters wide. Selectable 110/300/600 baud rates and prints at 60 cps. RS-232 interface. This daisy belt type printer is almost silent in operation. $175.
Bob Best,

OSI Series II converted to C4 with 64x32 video board, OZI expansion board (40k free RAM), CPU runs at 2 MHz, Compsoft C4 ROM, in case; 1 - MPI 51 disk drive, also in case (has space for 2 drives); 1 - National B&W TV for video. Software to suit. $700
Contact Bill Clarkson,

OSI C2-4P with 32K RAM, 540 video board, DAC, sound, speech synthesiser, serial RS 232, joystick and keypad ports, 16 pin I/O bus (compatable with Rabble CP/M unit), cassette - 300, 600 baud, disk controller (5" and 8" compatable), MPI B52 double sided disk drive, 1 MHz or 2 MHz switch; the latest DOS, includes PRINT AT, KEY SCAN/SCREEN SCAN etc. Visicalc, Budget, Database and lots of OSI adventures etc. On board power supply, (no transformer needed). Manuals for OS65D 4.0 and hundreds of hints, First Book of OSI and 12 months of 'Micro' $880 ono
Contact R. Thompson,

Registered by Australia Post
Publication No. VBG4212

If undeliverable return to
KAOS, 10 Forbes St
Essendon, Victoria 3040